REMARKS/ARGUMENTS

Claims 1-30 are pending in the present application. Claims 8 and 15-16 were amended. Reconsideration of the claims is respectfully requested.

I. 35 U.S.C. § 103, Obviousness

The Examiner has rejected claims 1-30 under 35 U.S.C. § 103(a) as being unpatentable over a document entitled "An Evaluation of Speculative Instruction Execution on Simultaneous Multithreaded Processors," a publication of Swanson et al. (hereinafter "Swanson"), in view of U.S. Patent Publication No. 2003/0182536, to Teruyama (hereinafter "Teruyama"). This rejection is respectfully traversed.

II. Teachings of Applicants

Applicants' invention relates to a method for issuing instructions in a multi-threaded microprocessor or the like that includes at least one multi-stage pipeline. As taught by Applicants, and as is well known by those of skill in the art, a pipeline processor has multiple stages that each instruction must traverse during processing, after it has been issued. At paragraphs [0004] and [0018], Applicants teach that "issue", as defined by their application, "means forwarding an instruction to the pipeline for processing".

In making their invention, Applicants were concerned about situations in which processed instructions caused stalls in a pipeline, which could result in the entire pipeline becoming stalled. Applicants recognized that a major cause of stalling is dependency, wherein processing of an instruction through a pipeline requires that a result must have been successfully achieved by a previous or prerequisite instruction. Otherwise, the dependent instruction would stall in the pipeline. To diminish stalling, Applicants position an instruction issue logic component between instruction input buffers and the pipeline processor. The instruction issue logic predicts the stage at which results of instructions will first become available for use or forwarding. The issue logic also identifies instructions as being dependent, and thus determines critical distance therefor. A critical distance is the number of pipeline stages between a stage when a dependent instruction will need a result, and the stage at which the result is made available by a prerequisite instruction. Based on the critical distance and other information, the instruction issue logic determines a confidence factor or probability that an instruction will complete all pipeline stages. The instruction issue logic then issues the instruction into the pipeline <u>only</u> if the confidence factor is above a predetermined threshold.

The above teachings of Applicants are set forth in the specification, such as at paragraphs [0006] and [0018] shown below:

> **[0006]** A method for issuing instructions in a simultaneous, multithreaded microprocessor. The microprocessor includes at least one multi-stage pipeline for processing instructions. The processor also includes a cache memory, multiple independent input buffers for each thread, and instruction issue logic. <u>The instruction issue logic is in a position between the input buffers and the multi-stage pipeline.</u> The method comprises the steps of receiving sets of instructions in the instruction issue logic at a rate equal to a clock rate of the microprocessor, wherein each set of instructions comprise one instruction from each of the independent input buffers. <u>The instruction issue logic then predicts the pipeline stage in which the results of the each instruction will be available for forwarding.</u> This pipeline stage information is then stored until the instruction reaches the predicted stage. <u>The issue logic then identifies any dependent instruction that requires a result from a preceding instruction that is within a critical distance of the dependent instruction.</u> The critical distance is equal to the number of stages between a stage when the dependent instruction will need the result and a stage when the result will be available. Based on the current contents of the pipeline, <u>the instruction issue logic issues all instructions that have a probability above a predetermined threshold</u> that the instruction will complete all stages of the pipeline without causing a stall. The instruction issue logic holds all instructions that have a probability below the predetermined threshold, wherein instructions held by the instruction issue logic are issued after their probability of not causing a stall in the pipeline rises above the threshold. **[Paragraph [0006] at page 4] (Emphasis added)**

> **[0018]** FIG. 2 is a more detailed diagram of multi-thread CPU 100. CPU input buffers 205 provide temporary storage for three instructions from each of the multiple threads of instructions. Of course, there can be more or less than three buffers for each instruction stream in other embodiments. The first column of buffers holds three instructions from the first thread of instructions. The second column of buffers holds three instructions from the second thread of instructions. This temporary storage system repeats for as many columns of buffers that are present on the CPU, i.e., until the n.sup.th column of buffers is filled with instructions from the n.sup.th thread. The present instruction issue method is flexible and can be used with any number of threads that a CPU is capable of accepting. <u>One instruction from each column of buffers enters instruction issue logic 200 where dependency problems for any instruction of that set are identified.</u> If an instruction is found to require an operand that will likely not be available when the instruction needs it, then the thread that the instruction came from is withdrawn from the pool of candidates for issuing, wherein <u>issuing means forwarding an instruction to the pipeline for processing</u>. The thread of instructions that is least likely to cause a stall in the pipeline is then issued. That is not to say that the entire thread is guaranteed to traverse the pipeline without interruption. The <u>instruction issue logic 200 evaluates a probability of causing a stall on every clock cycle in the preferred embodiment</u>. Thus, if an instruction from a thread that is currently being processed is found to have a high probability of causing a stall, that thread will be delayed and an instruction from another thread will issue. When an instruction is issued, it enters the first stage "A" of the shared pipeline stages 210. Different pipelines have different numbers of stages and FIG. 2 is drawn to show that the present instruction issue logic is compatible with pipelines having any number of stages. A different operation is performed on the instructions at each stage

of the pipeline 210. Stages A, B, C, W, X, Y, and Z are shared resources which may contain a mixture of instructions from different threads, however any one stage may contain only one instruction at any given time. The instruction issue logic 200 must choose appropriate instructions from the n threads to merge into the shared resources. In this example, instructions are required to resolve their dependencies before leaving stage B. In stage B, the operands on which the instruction is to operate are required. Stage Y represents the first opportunity in the pipeline that instruction results may be forwarded to dependent instructions in stage B. Thus, in this example, the pipeline has a "critical range" of four clock cycles, assuming there is a total of seven stages and each stage requires only one clock cycle. The critical range defines how long a dependent instruction must be delayed after issue of the instruction on which it depends before the dependent instruction can be issued without causing a stall in the pipeline. When the present method is applied to this example, an instruction with a dependency will be blocked from issuing until the instruction on which it depends reaches stage W. By delaying issue in this manner, the dependent instruction will reach stage B at the same time the instruction on which it depends reaches stage Y, assuming there are no pipeline holds. After issuing, the dependent instruction is able to proceed down the pipeline 210 without having to stall in any shared resources and having minimal impact on the execution of the other threads. This technique achieves the maximum multithreaded throughput. If issue of the dependent instruction were not delayed, the dependent instruction would reach stage B before the instruction on which it depends reached stage Y, causing the pipeline to stall and impacting the performance of all other threads. Blocking issue of the dependent instruction allows the other, unrelated threads, to use the "instruction slots" that otherwise would have been wasted by the dependent instruction blocking the pipe. **[Paragraph [0018] at pages 7-8] (Emphasis added)**

Claim 1 recites a useful embodiment of Applicants' invention, and reads as follows:

1. A method for issuing instructions in a multithreaded computer processor, the method comprising the steps of:

receiving sets of computer instructions in an instruction issue logic, wherein each set of instructions comprises one instruction from each of a plurality of independent instruction threads;

predicting a stage, within a multi-stage instruction pipeline of the computer processor, where results of each instruction will be available;

identifying as dependent instructions those received instructions that require a result from a prerequisite instruction;

determining a confidence factor for each received instruction that indicates a probability that the instruction will complete all stages of the pipeline without causing a stall; and,

issuing, from the instruction issue logic, instructions with confidence factors above a predetermined threshold.

III. Rejection of Claim 1

In the Office Action, the Examiner stated the following in rejecting Claim 1:

5. Referring to claims 1, 8, 16, and 24, taking claim 16 as exemplary, Swanson has taught a simultaneous multithreaded computer processor with speculative instruction issue that increases throughput, the computer processor comprising:

a. Multiple independent input buffers, wherein one set of buffers is provided for each of a plurality of independent threads of instructions (Swanson page 315, paragraphs 2-3; page 316, paragraph 2; page 319, paragraph 4; page 330, paragraph 1; and page 336, paragraph 4 to page 337, paragraph 1). In regards to Swanson, the individual buffers for each thread is inherent. Please see Parady, U.S. Patent Number 5,933,627 and Loikkanen and Bagherzadeh's "A Fine-Grain Multithreading Superscalar Architecture" ©1996 for more information.

b. Instruction issue logic that is connected to the independent input buffers (Swanson page 315, paragraphs 2-3; page 316, paragraph 2; page 319, paragraph 4; page 330, paragraph 1; and page 336, paragraph 4 to page 337, paragraph 1), wherein the instruction issue logic:

i. Receives instructions from each of the threads of instructions (Swanson page 315, paragraphs 2-3; page 316, paragraph 2; page 319, paragraph 4; page 330, paragraph 1; and page 336, paragraph 4 to page 337, paragraph 1);

ii. Determines a confidence factor for each instruction that indicates a probability that the instruction will complete all stages of the multi-stage pipeline without causing a stall (Swanson page 317, paragraph 1; page 326, paragraphs 4-7; page 327, paragraph 1 to page 328, paragraph 1; and page 336, paragraph 4 to page 337, paragraph 1); and,

iii. Issues instructions with confidence factors above a predetermined threshold (Swanson page 317, paragraph 1; page 326, paragraphs 4-7; page 327, paragraph 1 to page 328, paragraph 1; and page 336, paragraph 4 to page 337, paragraph 1).

6. Swanson has not taught

i. Predicts a stage, within a multi-stage pipeline of the processor, in which a result from each instruction will be available;

ii. Identifies as dependent instructions those received instructions that require a result from a prerequisite instruction; and

iii. Wherein a first stage of the multi-stage pipeline is connected to an output buffer of the instruction issue logic.

7. Teruyama has taught

i. Predicts a stage, within a multi-stage pipeline of the processor, in which a result from each instruction will be available (Teruyama paragraphs 0011-0014, 0032-0037, and 0082-0083; and Figure 5);

ii. Identifies as dependent instructions those received instructions that require a result from a prerequisite instruction (Teruyama paragraphs 0011-0014, 0032-0037, and 0082-0083; and Figure 5; and

iii. Wherein a first stage of the multi-stage pipeline is connected to an output buffer of the instruction issue logic (Teruyama paragraphs 0011-0014, 0032-0037, and 0082-0083; and Figure 5).

8. A person of ordinary skill in the art at the time the invention was made, and as taught by Teruyama, would have recognized that tracking the dependencies of an instruction ensures that instructions without dependencies are not cancelled inappropriately that results in reduced efficiency and speed (Teruyama paragraph 0013), thereby improving efficiency and speed of the processor. Therefore, it would have been

obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the dependency tracker of Teruyama in the device of Swamson to improve processor efficiency and speed.
9. Claims 1, 8, and 24 are functionally equivalent to claim 16 and are rejected for similar reasons. Claim 1 differs only in that it is a method claim. Claims 8 and 24 differ in that they are for a single threaded system. Swanson states on page 315, paragraph 2 that a simultaneous multithreading system can execute one thread and that all resources are dedicated to that thread, meaning that an SMT system can be a single threaded system with all buffers containing instructions for that thread. **|Office Action dated June 29, 2006 at pages 2-5|**

In order to establish a *prima facie* case of obviousness under 35 U.S.C. § 103, **MPEP § 2143** requires that there must be some suggestion or motivation in the prior art to modify the reference or to combine reference teachings as proposed, and the prior art or combined references must teach or suggest all the claim limitations. The suggestion to make the claim combination must be found in the prior art, not in the Applicant's disclosure. *In re Vacek*, 20 U.S.P.Q.2d 1438 (Fed. Cir. 1991). Moreover, in accordance with **MPEP § 2141.02**, each prior art reference must be considered in its entirety, i.e., as a whole, including portions that would lead away from the claimed invention. *W.L. Gore & Associates, Inc. v. Garlock, Inc.*, 220 U.S.P.Q. 303 (Fed. Cir. 1983). A third essential requirement for establishing a *prima facie* case, set forth in **MPEP § 2143.01**, is that the proposed modification cannot render the prior art unsatisfactory for its intended purpose.

In the present case, not all of the features of the claimed invention have been properly considered, and the teachings of the references themselves do not teach or suggest the claimed subject matter to a person of ordinary skill in the art. For example, no combination of Swanson and Teruyama teaches or suggests, in the over-all combination of Claim 1, either of the following Claim 1 features:

(1) Predicting a stage within a multi-stage instruction pipeline of a computer processor, where results of each instruction will be available (hereinafter "Feature (1)").

(2) Issuing instructions from the instruction issue logic that have a confidence factor above a predetermined threshold (hereinafter "Feature (2)").

IV. Feature (1) of Claim 1 Distinguishes over Both Cited References

Applicants consider that pertinent teachings of the Swanson reference are shown at page 315, paragraph 1; page 316, paragraph 2; the paragraph bridging pages 316-317; page 319, paragraph 4; page 327, paragraph 1 to page 328, paragraph 1; and the paragraph bridging pages 329-330. These sections of Swanson are as follows:

> Instruction speculation is a crucial component of modern superscalar processors. Speculation hides branch latencies and thereby boosts performance by executing the likely branch path without stalling. Branch predictors, which provide accuracies up to 96% (excluding OS code) [Gwennap 1995], are the key to effective speculation. The primary disadvantage of speculation is that some processor resources are invariably allocated to useless, wrong-path instructions that must be flushed from the pipeline. However, since resources on superscalars are often underutilized because of low single-thread instruction-level parallelism (ILP) [Tullsen et al. 1995; Cvetanovic and Kessler 2000], the benefit of speculation far outweighs this disadvantage and the decision to speculate as aggressively as possible is an easy one. **[Paragraph 1 on page 315] (Emphasis added)**

> We attempt to improve speculation performance on SMT by reducing wrong-path speculative instructions, either by not speculating at all or by using speculation-aware fetch policies (including policies that incorporate confidence estimators). To explain the results, we investigate which hardware structures and pipeline stages are affected by speculation, and how speculation on SMT processors differs from speculation on a traditional superscalar. Finally, we explore the boundaries of speculation's usefulness on SMT by varying the number of hardware threads, the number of functional units, and the cache capacities, and by using synthetic workloads to change the branch frequency and ILP within threads. **[Paragraph 2 on page 316] (Emphasis added)**

> The baseline configuration for our experiments is shown in Table I. For most experiments we used the ICOUNT fetch policy [Tullsen et al. 1996]. ICOUNT gives priority to threads with the fewest number of instructions in the pre-issue stages of the pipeline and fetches 8 instructions (or to the end of the cache line) from each of the two highest priority threads. From these instructions, it chooses to issue up to 8, selecting from the highest priority thread until a branch instruction is encountered, then taking the remainder from the second thread. In addition to ICOUNT, we also experimented with three alternative fetch policies. The first does not speculate at all, that is, instruction fetching for a particular thread stalls until the branch is resolved; instead, instructions are selected only from the non-speculative threads using ICOUNT. The second favors non-speculating threads by fetching instructions from threads whose next instructions are non-speculative before fetching from threads with speculative instructions. The third uses branch confidence estimators to favor threads with high-confidence branches. In all cases, ICOUNT breaks ties. **[Paragraph 5 beginning on page 316] (Emphasis added)**

> This section presents the results of our simulation experiments on instruction speculation for SMT. Our goal is to understand the trade-offs between two alternative means of hiding branch delays: instruction speculation and SMT's ability to execute instructions from multiple threads each cycle. First, we compare the performance of an SMT processor with and without speculation and analyze the differences between these two options. Then we discuss the impact of speculation-aware fetch policies and the use of branch prediction confidence estimators on speculation performance. **[Paragraph 4 on page 319] (Emphasis added)**

There are at least two different ways to use such confidence information. In the first, hard confidence, the processor stalls a thread on a low confidence branch, fetching from other threads until the branch is resolved. In the second, soft confidence, the processor assigns a fetch priority according to the confidence of a thread's most recent branch.

Hard confidence schemes use a confidence threshold to divide branches into high- and low-confidence groups. If the confidence value is above the threshold, the prediction is followed; otherwise, the issuing thread stalls until the branch is resolved. Hard confidence uses ICOUNT to select among the high confidence threads, so the confidence threshold controls how significantly ICOUNT affects fetch. Low thresholds leave the choice almost entirely to ICOUNT, because most threads will be high confidence. High thresholds reduce its influence by providing fewer threads from which to select.

Using hard confidence has two effects. First, it reduces the number of wrong-path-speculative instructions by keeping the processor from speculating on some incorrect predictions (i.e., true negatives). Second, it increases the number of correct predictions the processor ignores (false negatives).

Table III contains true and false negatives for the baseline SMT and an SMT with several hard confidence schemes when executing SPECINT95. Since our MacFarling branch predictor [McFarling 1993] has high accuracy (workload-dependent) predictions that range from 88% to 99%), the false negatives out-number the true negatives by between 3 and 6 times. Therefore, although mis-predictions declined by 14% to 88% (data not shown), this benefit was offset by lost successful speculation opportunities, and IPC never rose significantly. In the two cases when IPC did increase by a slim margin (less than 0.5%), JRS and Distance each with a threshold of 1, there were frequent ties among many contexts. Since ICOUNT breaks ties, these two schemes end up being quite similar to ICOUNT.

In contrast to hard confidence, the priority that soft confidence calculates is integrated into the fetch policy. We give priority to contexts that aren't speculating, followed by those fetching past a high confidence branch; ICOUNT breaks any ties. In evaluating soft confidence, we used the same three confidence estimators. Table IV contains the results for SPECINT95. From the table, we see that soft confidence estimators hurt performance, despite the fact that they reduced wrong-path-speculative instructions to between 0.1% and 9% of instructions fetched.

Overall, then, neither hard nor soft confidence estimators improved SMT performance, and actually reduced performance in most cases. **[Paragraph 1 on page 327 – Paragraph 1 on page 328] (Emphasis added)**

We draw two conclusions from this discussion. First, the key to speculation's benefit is its low cost compared to the benefit of the diverse thread mix it provides in the IQ. If branch prediction was less accurate, speculation would be more costly, and the diversity it adds would not compensate for resources wasted on mispeculation. However, as we will see in Figure 6, branch prediction accuracy generally has to be extremely poor to tip the balance against speculation.

Second, although we investigated only a few of the many conceivable speculation-aware fetch policies, there is little hope that a different speculation-aware fetch policy could improve performance. An effective policy would have to avoid significantly altering the distribution of fetched instructions among the threads and, simultaneously, significantly reduce the number of useless instructions fetched. Given the accuracy of modern predictors, devising such a mechanism is unlikely. **[Paragraph 2 on page 329 – paragraph 3 beginning on page 329 and ending on page 330] (Emphasis added)**

In the Office Action, the Examiner acknowledged that Swanson does not teach Applicants' Feature (1), that is, does not teach predicting a stage within a pipeline processor where results of each instruction will be available. Accordingly, the Examiner cited only Teruyama against this feature.

The Swanson reference evaluates the use of speculative techniques to execute instructions on simultaneous multithreaded (SMT) processors, which may use pipeline processing. However, Swanson emphasizes repeatedly that its disclosure is directed only to processing errors caused by wrong path speculative instructions. This is demonstrated, for example, by repeated reference to paths and branches in the above sections of Swanson. In regard to wrong path errors, processing of an instruction follows a path that can encounter one or more branches. Accordingly, Swanson discloses the use of confidence levels or confidence estimators to predict the probability that processing will follow the right path or branch, rather than a wrong path or unexpected branch.

It is readily apparent that since Swanson is exclusively concerned with wrong path errors, it provides no teaching in regard to processing of dependent instructions, or to problems in processing such instructions. Thus, Swanson clearly fails to show or suggest important teachings of Applicants' claims. As discussed above, the Office Action acknowledges that Swanson fails to teach Feature (1) of Claim 1.

Pertinent teachings of Teruyama are shown by the abstract, Figure 1, and claim 1 thereof, and at paragraphs [0011], [0014], [0041], [0083] and [0124]. These sections of Teruyama are as follows:

> **[0011]** In the aforementioned document, it is predicted whether or not the load instruction has hit. Only when it is predicted that the load instruction has hit, the dependent instruction is executed. The probability of canceling an instruction is thereby lowered. However, even when it is predicted that the load instruction has hit and an instruction not dependent on the load instruction is issued, there are cases where the load instruction has actually not hit. In this case, the instruction not dependent on the load instruction is needlessly cancelled. **(Emphasis added)**
>
> **[0014]** According to an aspect of the invention, there is provided an instruction issuing device comprising: an instruction issuing section which speculatively issues instructions out-of-order; a first detecting circuit which detects direct dependencies between the instructions issued from the instruction issuing section and a plurality of instructions including a load instruction in each stage of a pipeline; and a second detecting circuit to which output signals of the first detecting circuit and cache miss signals of the load instruction are supplied, the second detecting circuit detecting indirect dependencies between the instructions issued from the instruction issuing section and the load instruction which cache-missed in each stage of the pipeline, on the basis of the output

signals of the first detecting circuit and the cache miss signals of the load instruction. **(Emphasis added)**

[0041] The load store unit 18b maintains data dependency via a memory for a load instruction and a store instruction processed out-of-order in a processor carrying out out-of-order execution. Concretely, the load store unit 18b grasps the order of the memory access instructions, and manages the order of the memory access instructions issued out-of-order. Further, when a data cache miss-hits in the execution of a load instruction, the load store unit 18b outputs a cache miss signal LOMiss1n (n is the stage of the pipeline). The cache miss signal LOMiss1n is supplied to the DLC 16. **(Emphasis added)**

[0083] Moreover, the DLC 16 is connected to each entry of the instruction window buffer 14a. The DLC 16 retrieves an instruction depending on the load instruction in accordance with a cache miss signal outputted from the load store unit 18b. A signal Depend1A showing dependency and outputted from the DLC 16 is supplied to the register score board unit 15a and the RAT 22. When the signal Depend1A is outputted from the DLC 16, the entry of the RAT 22 for the dependent physical register is invalidated on the basis of the status of the instruction of the register score board unit 15a. Moreover, the update circuit 21 resets the dependent physical register in an invalid state in the instruction window buffer 14a. The detailed operation when a cache miss arises at the time of executing the load instruction will be described later. **(Emphasis added)**

[0124] FIG. 12 shows an embodiment of the DLC 16. In FIG. 12, a first detecting circuit 16a detects a register depending on the load instruction directly. Further, a second detecting circuit 16b detects indirect dependencies of plural stages.

A first detecting circuit detects a register depending directly on a load instruction. A second detecting circuit detects indirect dependencies of plural stages between all instructions in a state of execution and all load instructions of the respective stages of a pipeline, in accordance with cache miss signals and output signals of the first detecting circuit. **[Abstract] (Emphasis added)**

1. An instruction issuing device comprising:

an instruction issuing section which speculatively issues instructions out-of-order;

a first detecting circuit which detects direct dependencies between the instructions issued from the instruction issuing section and a plurality of instructions including a load instruction in each stage of a pipeline; and

a second detecting circuit to which output signals of the first detecting circuit and cache miss signals of the load instruction are supplied, the second detecting circuit detecting indirect dependencies between the instructions issued from the instruction issuing section and the load instruction which cache-missed in each stage of the pipeline, on the basis of the output signals of the first detecting circuit and the cache miss signals of the load instruction. **[Claim 1] (Emphasis added)**

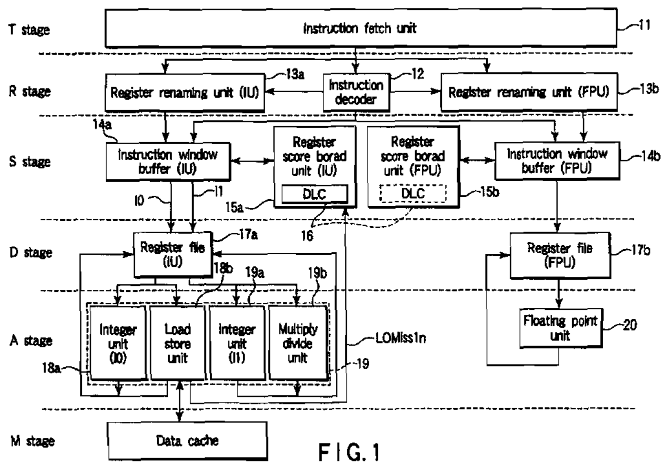

FIG. 1

From the above sections, such as the abstract, paragraph [0014] and claim 1, it is seen that Teruyama discloses an arrangement wherein a first circuit detects direct dependencies between issued instructions and a load instruction in each stage of a pipeline processor. In addition, a second circuit detects indirect dependencies between issued instructions and a load instruction which has cache-missed. As taught at paragraph [0041], a cache miss occurs when data is not loaded into a cache as required by a load instruction. When a data cache miss occurs, load store unit 18b outputs a cache miss signal, which is supplied to DLC 16. Paragraph [0124] discloses that DLC 16 comprises first and second detecting circuits 16a and 16b, respectively.

In view of the above teachings, it is clear that Teruyama eschews or opposes Applicants' Feature (1), that is, predicting a stage in an instruction pipeline where results of respective instructions will be available. To the contrary, Teruyama stresses that its arrangement deliberately avoids predicting the availability of any instruction results, such as a cache hit or miss resulting from a load instruction. Instead, Terayuma proceeds to issue and execute successive instructions, until a cache miss occurs. This event then triggers operation of Teruyama's DLC 16, to detect dependencies between instructions being executed and the load instructions in the pipeline.

At paragraph [0011], Teruyama teaches that predictions as recited by Applicants' Feature (1) are in fact undesirable. More particularly, Teruyama states that if it is predicted that a load instruction has hit, but actually has not hit, an instruction not dependent on the load instruction may be "needlessly canceled." Thus, Teruyama emphasizes the principle that, as an alternative to predicting results of instructions, respective instructions will simply be executed, until there is notice of a problem. Clearly, this emphasis of Teruyama diametrically teaches away from the predicting Feature (1) of Applicants' Claim 1.

V. Feature (2) of Claim 1 Distinguishes over Both Cited References

In discussing Claim 16 in the Office Action, the Examiner stated that the final element of Claim 16 is not taught by Swanson. This element recites connecting a first stage of the multi-stage pipeline to an output buffer of the instruction issue logic. The Examiner also stated that Claim 16 is functionally equivalent to Claim 1. In Claim 1, the element thereof that is equivalent to the final element of claim 16 is the portion of Feature (2) directed to issuing instructions from the instruction issue logic. Thus, this portion of Feature (2) is clearly not taught by Swanson.

Moreover, Swanson fails to show Feature (2) of Applicants' Claim 1 in its entirety. Feature (2) requires issuing instructions from the instruction issue logic that have a confidence factor above a predetermined threshold. Swanson does discuss executing speculative instructions on SMT processors. However, nowhere does Swanson appear to teach a mechanism for issuing instructions that is equivalent to the instruction issue logic of Feature (2) of Claim 1.

Teruyama likewise fails to disclose the instruction issue logic of Feature (2). In reciting the issuing of instructions, Feature (2) requires that instructions be directed into a pipeline processor from the instruction issue logic. However, as exemplified by its Figure 1, Teruyama discloses no components or structures that precede instruction fetch unit 11, at stage T. Stage T is clearly the first stage of the pipeline processor of Teruyama, and all lower stages shown in Figure 1 are further stages of the pipeline. Circuits discussed in Teruyama, such as DLC 16, are all contained within the pipeline, and are thus not available to issue instructions to the pipeline, as recited by Applicants' Feature (2).

Moreover, as discussed above, Teruyama teaches away from issuing instructions that are based on predictions or have confidence factors, as likewise recited by Feature (2).

VI. No Basis for Combining References to Reject Claim 1

In order to reject Claim 1 for obviousness under 35 U.S.C. § 103, by combining Swanson and Teruyama as proposed by the Examiner, the Examiner must first demonstrate that there is some basis or motivation in the prior art for making the combination. Applicants consider that the Examiner has not done this in the Office Action. However, courts continue to hold that it is absolutely essential to provide such prior art motivation, in order to establish a *prima facie* case of obviousness by combining references. For example, in an opinion of the U.S. Court of Appeals for the Federal Circuit (CAFC), decided in January, 2005, the Court stated the following:

> "Our case law makes clear that the best defense against the subtle but powerful attraction of a hindsight-based obviousness analysis is rigorous application of the requirement for a showing of the teaching or motivation to combine prior art references." *Dembiczak, 175 F.3d at 999*; see also *Ruiz, 234 F.3d at 665* (explaining that the temptation to engage in impermissible hindsight is especially strong with seemingly simple mechanical inventions). This is because "combining prior art references without evidence of such a suggestion, teaching, or motivation simply takes the inventor's disclosure as a blueprint for piecing together the prior art to defeat patentability--the essence of hindsight." [*286] *Dembiczak, 175 F.3d at 999*. Therefore, we have consistently held that a person [**9] of ordinary skill in the art must not only have had some motivation to combine the prior art teachings, but some motivation to combine the prior art teachings in the particular manner claimed. See, e.g., *In re Kotzab, 217 F.3d 1365, 1371 (Fed. Cir. 2000)* ("Particular findings must be made as to the reason the skilled artisan, with no knowledge of the claimed invention, would have selected these components for combination in the manner claimed." (emphasis added)); *In re Rouffet, 149 F.3d 1350, 1357 (Fed. Cir. 1998)* ("In other words, the examiner must show reasons that the skilled artisan, confronted with the same problems as the inventor and with no knowledge of the claimed invention, would select the elements from the cited prior art references for combination in the manner claimed." (emphasis added)). *Teleflex, Inc. v. KSR Int'l Co.*, 119 Fed. Appx. 282, 285-286 (Fed. Cir. 2005).

In the Office Action, no basis was provided for combining Swanson with Teruyama to reject Claim 1, except for an unsupported reason as to why such combination would be obvious to a person of ordinary skill in the art. Applicants consider that this unsupported reason is clearly insufficient to meet the motivational requirements of *Teleflex, Incorporated v. KSR International Co.*, set forth above, in order to allow combination of references. Moreover, Applicants consider that they are entitled to challenge the alleged basis or motivation for combining references against their claims. To do this, Applicants must have a citation to a prior art reference that teaches the reason for the combination. This is necessary to provide a clear and complete picture of what was actually done previously by one of skill in the art, not an opinion as to what one of skill in the art might do.

The *Teleflex v. KSR* opinion emphasizes that "the requirement for a showing of the teaching or motivation to combine prior art references" is particularly important when considering inventions that appear to be simple. The opinion states that for simple inventions, "the temptation to engage in impermissible hindsight is especially strong." Applicants consider that the recitation of their Claim 1, because it is clear and easy to understand, may be considered "simple." Accordingly, in view of the above opinion of the CAFC, Claim 1 cannot be rejected by combining Swanson and Terayuma without a "rigorous application" of the requirement to teach a motivation for such combination.

Applicants' Claim 1 recites five steps, that is, the steps of receiving, predicting, identifying, determining and issuing. In the Office Action, the Examiner acknowledged that the Swanson reference does <u>not</u> teach the predicting, identifying or issuing steps, or at least not a portion of the issuing step, as discussed above. Thus, Swanson fails to disclose at least <u>half</u> of the recited elements of Claim 1. As a result, Swanson would require substantial modification, in order to realize Applicants' Claim 1. One of skill in the art therefore would <u>not</u> be motivated to undertake any such modification, without <u>clear guidance</u> to do so that was <u>shown by the prior art</u>. As discussed above, Applicants have been provided with no such guidance or motivation.

In regard to Teruyama, an essential principle thereof as discussed above, is to execute successive instructions <u>until</u> a problem such as a missed cache appears. Accordingly, Teruyama is not concerned with <u>speculative</u> processing of instructions, and thus effectively <u>teaches away</u> from any combination with Swanson, or other reference, to realize key elements of Applicants' Claim 1. Such elements include predicting availability of instruction results, determining a confidence factor, and issuing instructions with confidence factors above a predetermined threshold.

VII. <u>Independent Claims 8, 16 and 24 Distinguish over Cited References</u>

Independent Claims 8, 16 and 24 respectively recite subject matter similar to the patenable subject matter of Claim 1, and are each considered to distinguish over the art, including the Swanson and Teruyama references, for the same reasons given in support thereof.

Claims 8 and 16 are additionally considered to patentably distinguish over the art in reciting the step of determining the confidence factor for a given dependent instruction by calculating a critical distance between the given dependent instruction and its corresponding prerequisite instruction, and in further reciting the definition of critical distance. This feature is disclosed in Applicants' specification, for example, at page 4, paragraph [0006]. Applicants consider that neither Swanson nor Teruyama, or any combination thereof, shows or suggests this feature.

VIII. <u>Remaining Claims Distinguish over Cited References</u>

Claims 2-7, 9-15, 17-23 and 25-30 depend on independent Claims 1, 8, 16 and 24, respectively, and are each considered to distinguish over the art for the same reasons given in support thereof.

Claim 3 is additionally considered to distinguish over the art in reciting the feature that the confidence factor for instruction is determined based upon a current location and predicted stage of the prerequisite instruction. Neither Swanson nor Teruyama, nor any combination thereof, shows or suggests this feature.

Claim 4 is additionally considered to distinguish over the art in reciting the step of dynamically recalculating the confidence factor for each instruction based on the current contents of the pipeline. Neither Swanson nor Teruyama, nor any combination thereof, shows or suggests this feature.

Claim 6 is considered to distinguish over the art in reciting the feature that the confidence factor for a dependent instruction is determined based upon the current location and the predicted stage of the prerequisite instruction and upon a predicted resolution of any identified shared resource conflict. Neither Swanson nor Teruyama, nor any combination thereof, shows or suggests this feature.

Claim 7 is additional considered to distinguish over the art in reciting the feature of dynamically recalculating the confidence factor for each instruction based on the current contents of the pipeline and a current status of any shared resources. Neither Swanson nor Teruyama, nor any combination thereof, shows or suggests this feature.

IX. Conclusion

It is respectfully urged that the subject application is patentable over both the Swanson and Teruyama references and any combination thereof, and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: September 29, 2006

Respectfully submitted,

/James O. Skarsten/

James O. Skarsten
Reg. No. 28,346
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants